MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM

Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, Stefan Leutenegger

***Abstract*— We propose a new multi-instance dynamic RGB-D SLAM system using an object-level octree-based volumetric representation. It can provide robust camera tracking in dynamic environments and at the same time, continuously estimate geometric, semantic, and motion properties for arbitrary objects in the scene. For each incoming frame, we perform instance segmentation to detect objects and refine mask boundaries using geometric and motion information. Meanwhile, we estimate the pose of each existing moving object using an object-oriented tracking method and robustly track the camera pose against the static scene. Based on the estimated camera pose and object poses, we associate segmented masks with existing models and incrementally fuse corresponding colour, depth, semantic, and foreground object probabilities into each object model. In contrast to existing approaches, our system is the first system to generate an object-level dynamic volumetric map from a single RGB-D camera, which can be used directly for robotic tasks. Our method can run at 2-3 Hz on a CPU, excluding the instance segmentation part. We demonstrate its effectiveness by quantitatively and qualitatively testing it on both synthetic and real-world sequences.**

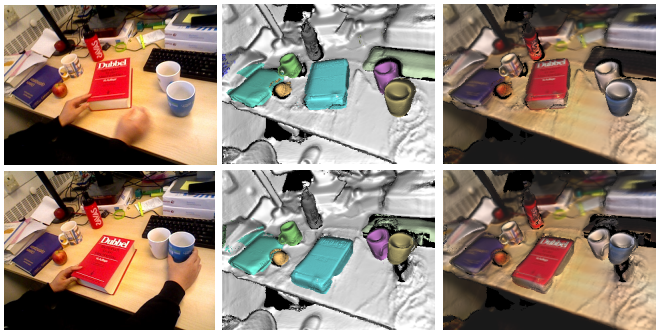
I. Introduction

In Simultaneous Localisation and Mapping (SLAM) both, the map of the unknown environment as well as the robot pose within it, are concurrently estimated from the data of its on-board sensors only. In recent years, the field of SLAM has experienced rapid progress. It started from sparse SLAM [1], [2], and evolved into dense SLAM [3] thanks to the increased computational power of GPU and affordability of depth sensors. More recently, many people have begun to leverage Deep Neural Networks and their ability to learn from large amounts of training data to improve SLAM. This fast-evolving research in SLAM has, since then, lead to strong progress in various fields of applications, such as robotics, Virtual Reality (VR), and Augmented Reality (AR).

Despite this progress, much work is still based on the fundamental assumption of a static environment, within which points in the 3D world always maintain the same spatial position in the global world, with the only moving object being the camera. This assumption enabled the success of early phases of development as it alleviated the chicken-and-egg problem between map estimation and sensor pose estimation. A camera pose can be estimated between a live frame and a reference frame, which is based on the assumption that the relative transformation between those two images is caused only by the camera motion. It is this basic, yet strong, assumption that allowed a joint probabilistic inference (sparse SLAM [4]) or an alternating optimisation (dense SLAM [5]) of map and pose relationship to solve SLAM. Any moving objects in the environment would be treated as outliers to the static model and are intentionally ignored by tracking and mapping.

The authors are with Department of Computing, Imperial College London, United Kingdom. {b.xu17, wenbin.li, dt214, m.bloesch, a.davison, s.leutenegger}@imperial.ac.uk

Video link: https://youtu.be/gturboNl9gg



Input Label Reconstruction

Fig. 1. An overview of our system. Given RGB-D images, our system builds an object-level dense volumetric map that deals with dynamic objects and ignores people. Next to the input image we show the labelled object models as well as the coloured reconstruction.

This idealised setup, therefore, can only handle a small amount of dynamic elements and disqualifies itself from many real-world applications as environments, especially where humans are present, change constantly. A robust SLAM system, which works in highly dynamic environments, is still an open problem, which we seek to address in this work.

Although dynamic SLAM has been studied for a couple of decades [6], approaches based on visual dense SLAM have only recently been explored. They can be categorised into three main directions. One deforms the whole world in a non-rigid manner in order to include a deformable/moving object [7]. The second specifically aims at building a single static background model, while ignoring all possibly moving objects and thus improving the accuracy of camera tracking [8], [9], [10], [11]. The third models dynamic components by creating sub-maps for every possibly rigidly moving object in the scene while fusing corresponding information into these sub-maps [12], [13], [14]. We are more interested in the third direction since we believe that, similar to human perception, an awareness of instances in the map would be a more proper solution for robots to perceive the changing environment and has higher potential to achieve a meaningful map representation. However, most existing approaches build maps using a collection of surfels, which is difficult to be used directly for robotic tasks. The only two systems that support sub-map volumetric map, we know of so far, are [13] and [15]. However, the former has been specifically designed for an outdoor stereo camera setting and the latter only

deals with static environments. Here, we propose the first object-level dynamic volumetric map for indoor environment applications, where free space and surface connectivity can be represented for each object model. We further improve its memory efficiency by utilising an octree-based structure. Despite showing some promising results based on deep learning, most methods [12], [14], [13] simply leverage predictions from neural network without much refinement in the map fusion. In this paper, we integrate and refine semantic predictions by fusing them into object models.

The main contributions in this paper are divided into four main parts. We propose

1) the first RGB-D multi-instance dynamic SLAM system using a volumetric representation,
2) a more robust tracking method utilising weighting via measurement uncertainty and being re-parametrised for object tracking,
3) an integrated segmentation using geometric, photometric, and semantic information,
4) a probabilistic fusion of semantic distribution and a foreground object probability into octree-based object models.

II. Related work

In the majority of SLAM systems the environment is assumed to be static. To tackle dynamic environment in real-world applications, several solutions have recently been proposed and they can be mainly categorised into three directions as introduced in last section. We will introduce and compare the last two types of approaches in further details in this section. One straightforward way for dynamic SLAM is to segment dynamic objects out as outliers and intentionally ignore them from tracking and reconstruction to avoid corruption in the pose estimation. StaticFusion [9] performs segmentation by coupling camera motion residuals, depth inconsistency and a regularisation term. Barnes *et al.* [10] learn to segment possibly moving objects in a self-supervised way, which is limited by the availability of training data and may often misclassify static objects. Bescos *et al.* [11] combine Mask-RCNN [16] with depth inconsistency checking to segment moving objects and further inpaint those areas with static background. Those methods provide a more robust approach in dynamic scene than conventional SLAM methods, however, information regarding the moving objects is lost. Instead, our approach aims to simultaneously track and reconstruct static background and dynamic and static objects in the scene, while at the same time, provide state-of-the-art tracking accuracy.

There are three approaches, to our knowledge, which provide similar functionality as ours and can reconstruct multiple moving objects in the scene – the third way to tackle dynamic SLAM. Co-Fusion [12] segments objects by either ICP motion segmentation or semantic segmentation and then tracks objects separately based on ElasticFusion [17]. MaskFusion [14] segments objects using a combination of instance segmentation from Mask-RCNN and geometric edges, and tracks objects using the same approach as Co-Fusion. Both Co-Fusion and MaskFusion use surfels to represent map models, which is memory efficient but cannot directly provide free space information in the map, and neither surface connectivity. DynSLAM [13] focuses on outdoor environments using stereo cameras. In contrast, our system focuses on indoor environments consisting of many (potentially) moving objects using a single RGB-D camera.

In terms of differences in system components, our system further differentiates itself from above approaches. In camera tracking, we weighted photometric and geometric terms by their measurement uncertainty, instead of a single weight such as in [17]. Also, to be robust to depth loss, we derive two terms from different frames to complement one another. To track objects, all previous methods use a virtual camera pose, which is not robust to object rotation due to its difficulty to converge. We found best robustness by re-parametrising it into object coordinate. To generate object masks, we combine both information to provide better boundary conditions, instead of using just motion or just semantic information. When fusing information to object models, we fuse not only depth and colour information, but also semantic and foreground predictions while previous methods just take predictions from neural network without any refinement. In terms of speed, all above three requires one even two powerful GPUs, while our method, despite running only on CPU, is capable of performing a similar speed to DynSLAM [13].

Another very recent work related to ours is Fusion++[15], which generates an object-level volumetric map yet in static environments. In addition to handling dynamic scenes, our system utilises a joint photometric and geometric tracking to robustly track both camera and object poses while Fusion++ only use geometric tracking to estimate camera pose. Furthermore, to have a better object mask boundary for fusion and tracking, we combine geometric, motion and existing model information to refine mask boundary instead of directly using predicted mask as was done in Fusion++. In terms of map representation, Fusion++ is based on discrete voxel grids, which suffers from scalability issues, while we represent all our object models in memory-efficient octree structures.

III. Notations and preliminaries

In this paper, we will use the following notation: a reference coordinate frame is denoted $\underrightarrow{\mathcal{F}}_A$. The homogeneous transformation from $\underrightarrow{\mathcal{F}}_B$ to $\underrightarrow{\mathcal{F}}_A$ is denoted as $\boldsymbol{T}_{AB}$, which is composed of a rotation matrix $\mathbf{C}_{AB}$ and a translation vector ${}_A\mathbf{r}_{AB}$. For each pair of images, we distinguish them as live (L) and reference (R) image. For example, a live RGB-D image contains the intensity image I_L and depth image D_L, with 2D pixel positions denoted as $\boldsymbol{u}_L$ and pixel lookup (including bilinear interpolation) denoted as $[\cdot]$. Perspective projection and back-projection are denoted π and π^{-1}, respectively.

In our system, we store every detected object into a separate object coordinate frame $\underrightarrow{\mathcal{F}}_{\mathbf{O}_n}$, with $n \in \{0\ldots,N\}$ where N is the total number of objects (excluding background) and 0 denotes background. We assume a canonical

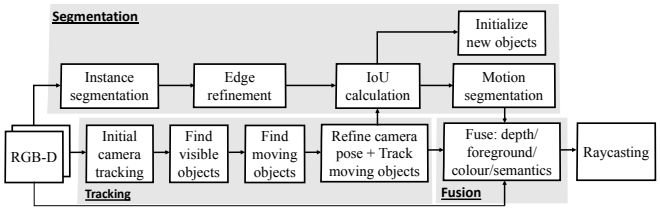


Fig. 2. The pipeline of the proposed method

static volumetric model is stored in each object coordinate frame, forming the basis of our multi-instance SLAM system. In addition, each object is also associated with a COCO dataset [18] semantic class label $c_n \in \{0,\dots,80\}$, a probability distribution over its potential semantic class labels, a current pose w.r.t. the world coordinate $\boldsymbol{T}_{WO_n}$, and a binary label $s \in \{0,1\}$ denoting whether the object is believed to be in motion or not. Each object is represented in an separate octree structure, where every voxel stores Signed Distance Function (SDF) value, intensity, foreground probability and the corresponding weights.

IV. Method

A. System overview

Fig. 2 shows the pipeline of our proposed system. It is composed of four parts: segmentation, tracking, fusion and raycasting. Each input RGB-D image is processed by Mask R-CNN to perform instance segmentation, which is followed by geometric edge segmentation and motion residuals from tracking to refine mask boundaries (Section IV-D). For the tracking, we first track the camera against all vertices excluding the human mask area (Section IV-B) and then raycast from this pose to find which objects are currently visible in this frame. This can also help associating local object masks with existing object models. We evaluate motion residuals for each object to determine if it is in motion or not, then track moving objects (Section IV-C) and refine the camera pose against the static world – which includes currently static objects (Section IV-B). Using estimated poses of the camera and objects, depth and colour information, as well as predicted semantic and foreground probabilities are fused into the object models (Section IV-E). Detection of visible objects as well as raycasting is explained in Section IV-F.

B. RGB-D Camera tracking

This part estimates the live camera pose $\boldsymbol{T}_{WC_L}$ and is composed of two steps. First, it tracks against all model vertices while masking out detected people; second, it tracks against all static scene parts. Both steps are conducted by minimising the dense point-to-plane ICP residual e_{g} and photometric (RGB) residual e_{p}, which are weighted by individual measurement uncertainty, w_{g} and w_{p}.

$$E_{\mathrm{track}}(\boldsymbol{T}_{WC_L}) = \frac{1}{2}\left(\sum_{\boldsymbol{u}_L \in M_L} w_{\mathrm{g}}\,\rho(e_{\mathrm{g}}) + \sum_{\boldsymbol{u}_R \in M_R} w_{\mathrm{p}}\,\rho(e_{\mathrm{p}})\right), \tag{1}$$

where ρ represents the Cauchy loss function and M is a mask excluding invalid correspondences (for ICP), occlusions (for RGB), and humans.

For the ICP residual, we use the method proposed in [3] to minimise point-plane depth error between the live depth map and the rendered depth map of the model on the reference frame:

$$e_{\mathrm{g}}(\boldsymbol{T}_{WC_L}) = {}_W\mathbf{n}^r[\boldsymbol{u}_R] \cdot (\boldsymbol{T}_{WC_L}\,{}_{C_L}\mathbf{v}[\boldsymbol{u}_L] - {}_W\mathbf{v}^r[\boldsymbol{u}_R]), \tag{2}$$

where ${}_{C_L}\mathbf{v}$ is live vertex map in the camera coordinate by back-projection and ${}_W\mathbf{v}^r$ and ${}_W\mathbf{n}^r$ are the rendered vertex map and normal map expressed in world coordinates. For each pixel $\boldsymbol{u}_L$ on the live depth map, its correspondence $\boldsymbol{u}_R$ on the rendered depth map can be found using projective data association:

$$\boldsymbol{u}_R = \pi(\boldsymbol{T}_{WC_R}^{-1}\,\boldsymbol{T}_{WC_L}\,(\pi^{-1}(\boldsymbol{u}_L, D_L[\boldsymbol{u}_L]))), \tag{3}$$

where $\boldsymbol{T}_{WC_R}$ is the camera pose of the reference frame.

For maximum robustness, we combine the ICP residual with a photometric one by rendering a depth map from model in the reference frame and using that depth map to align photometric consistency:

$$e_{\mathrm{p}}(\boldsymbol{T}_{WC_L}) = I_R[\boldsymbol{u}_R] - I_L[\pi(\boldsymbol{T}_{WC_L}^{-1}\,(\boldsymbol{T}_{WC_R}\,\pi^{-1}(\boldsymbol{u}_R, D_R^r[\boldsymbol{u}_R])))]. \tag{4}$$

Different from previous approaches [12], we evaluate the photometric residuals using rendered reference depth map other than the raw depth map on live frame or reference frame for the de-noised depth quality from models. This choice furthermore improves the robustness of tracking when raw input depth is not available, e.g. when the camera is too close to a surface.

We further introduce a measurement uncertainty weight to combine ICP and RGB residuals. For RGB residuals, the measurement uncertainty is assumed to be constant for all pixels. For ICP residuals, the quality of input depth map is related to the structure of the depth sensor and the depth range. We adopted the inverse covariance definition for depth measurement uncertainty in [19]. Given the sensor parameters, i.e. baseline b, disparity d, focal length f, and the uncertainties in the x-y plane σ_{xy} and disparity direction σ_z, the standard deviation σ_D for depth sensor measurement in the x, y, z coordinates can be modelled as:

$$\boldsymbol{\sigma}_D = \left(\frac{D_L[\boldsymbol{u}_L]}{f}\sigma_{xy}, \frac{D_L[\boldsymbol{u}_L]}{f}\sigma_{xy}, \frac{D_L^2[\boldsymbol{u}_L]}{fb}\sigma_z\right). \tag{5}$$

The weight for ICP residuals using the inverse covariance of measurement uncertainty is then defined as:

$$w_{\mathrm{g}} = \frac{1}{({}_W\mathbf{n}^r)^T\,{}_W\mathbf{n}^r\,\boldsymbol{\sigma}_D^T\boldsymbol{\sigma}_D}. \tag{6}$$

The cost function is minimised using the Gauss-Newton approach in a three-level coarse-to-fine scheme. The necessary Jacobians are omitted for space constraint.

After performing an initial camera tracking, we raycast to find visible objects in the view. To find which objects are in motion, we evaluate $E_{\mathrm{track}}(\boldsymbol{T}_{WC_L})$ once again on the finest

level on the *live* frame. To this end, the RGB residual needs to be re-formulated as:

$$e_{\mathrm{p}}(\boldsymbol{T}_{WC_L}) = I_L[\boldsymbol{u}_L] - I_R[\pi(\boldsymbol{T}_{WC_R}^{-1}(\boldsymbol{T}_{WC_L}\pi^{-1}(\boldsymbol{u}_L, D_L^r[\boldsymbol{u}_L])))]. \tag{7}$$

We apply a threshold to the combined residual $E_{\mathrm{track}}(\boldsymbol{T}_{WC_L})$ to find the motion inliers. If the inlier ratio is lower than 0.9 in the object's rendered mask, then we consider that object is moving and refine its pose as described in Section IV-C. The camera pose is then refined by tracking against only static objects using the same objective function and optimisation strategy explained above.

C. Object pose estimation

In this part, we describe how to estimate the pose of moving objects. As opposed to virtual camera based tracking [12], [10], we propose to employ an object-centric approach, which is less prone to bad initial pose guesses. We still use a joint dense ICP and RGB tracking, weighted in the same way as Eq. 1, just with different ICP and RGB residual definitions. In the present formulation, we estimate the current relative pose between object and camera, $\boldsymbol{T}_{C_LO_L}$, by aligning the live vertex map expressed in the live object frame with the rendered vertex map expressed in the reference object frame:

$$e_{\mathrm{g}}(\boldsymbol{T}_{C_LO_L}) = \mathbf{C}_{WO_R}^{-1}\,{}_W\mathbf{n}^r[\boldsymbol{u}_R]\cdot(\boldsymbol{T}_{C_LO_L}^{-1}\,{}_{C_L}\mathbf{v}[\boldsymbol{u}_L] - \boldsymbol{T}_{WO_R}^{-1}\,{}_W\mathbf{v}^r[\boldsymbol{u}_R]). \tag{8}$$

The formulation is based on the assumption that each object coordinate frame yields a static canonical object model and thus the point clouds must align. The proposed parameterisation leads to more stable tracking due to a smaller lever arm effect of the rotation. When computing the partial derivative of the above cost w.r.t. the rotation [20] we get a term proportional to $\mathbf{C}_{C_LO_L}^{-1}({}_{C_L}\mathbf{v}[\boldsymbol{u}_L] - {}_{C_L}\mathbf{r}_{C_LO_L})$ which is small since we choose the object frame to be centred. In analogy, we also re-formulate the RGB residual as:

$$e_{\mathrm{p}}(\boldsymbol{T}_{C_LO_L}) = I_R[\boldsymbol{u}_R] - I_L[\pi(\boldsymbol{T}_{C_LO_L}\boldsymbol{T}_{C_RO_R}^{-1}(\pi^{-1}(\boldsymbol{u}_R, D_R^r[\boldsymbol{u}_R])))]. \tag{9}$$

The above cost function is also optimised using Gauss-Newton approach in a three-level coarse-to-fine scheme with $\boldsymbol{T}_{C_LO_L}$ initialised as $\boldsymbol{T}_{C_LO_R}$.

D. Combined semantic-geometric-motion segmentation

For each RGB-D frame, we use Mask R-CNN [16] to find semantic instances, followed by geometric edge refinement to solve leaked mask boundary [14]. Then we render instance masks for each map object to the live frame by means of raycasting. We associate local segmentation masks, which are generated from Mask R-CNN and geometric refinement, with existing object models by calculating the intersection of union (IoU) with the rendered masks. We assign the local segmentation mask to the rendered mask which has the largest intersection and where the intersection is larger than 0.5. In comparison to [14], we do not require predicted semantic label of the local segmentation mask to be the same as object semantic class since the prediction may be subject to high uncertainty. Instead, we trust probabilistic fusion of semantic predictions to refine the objects' semantic labels (described in Section IV-E).

For segmentation masks that do not belong to any existing objects, a new object model will be initialised (described in Section IV-E). For objects without associated local segmentation masks, i.e. Mask R-CNN has no corresponding detection, we choose its rendered mask from the model for the subsequent fusion process.

After associating segmentation masks with object models, we further refine the segmentation masks based on motion residuals of object tracking. We evaluate Eq. (1) again on the finest level, however, this time we evaluate photometric residual on the live frame:

$$e_{\mathrm{p}}(\boldsymbol{T}_{WC_L}) = I_L[\boldsymbol{u}_L] - I_R[\pi(\boldsymbol{T}_{C_RO_R}\boldsymbol{T}_{C_LO_L}^{-1}(\pi^{-1}(\boldsymbol{u}_L, D_L^r[\boldsymbol{u}_L])))]. \tag{10}$$

Pixels whose joint ICP and RGB residuals are too high are treated as outliers and filtered out in the segmentation mask.

Before integration, we also generate a foreground mask based on the local segmentation mask. The use of foreground probabilities is inspired by the foreground/background probabilities introduced in [15] and allows to avoid spurious integration due to wrong segmentation masks. Information in both foreground and background regions are integrated into the models. In order to avoid impairing the efficiency of the octree structure, we use dilated segmentation masks as background mask. Pixels in the foreground are assigned an foreground probability of 1.0 while pixels in the dilated background are assigned 0. For undetected existing objects that Mask R-CNN fails on, we assign an foreground probability of 0.5 to their foreground due to their lower possibility of existence.

E. Object-level fusion

From each frame, we integrate depth, colour, semantics and foreground probability information into object models using foreground and background masks. Using the relative pose $\boldsymbol{T}_{O_nC_L}$ and depth, the Truncated SDF (TSDF) is updated following the approach of Vespa et al. [21]. Concurrently within the same voxels, colour and foreground probability are updated using a weighted average. For semantic fusion, we refine the semantic class probability distribution for each model using averaging, instead of Bayesian updating which often leads to overconfidence when used with Mask R-CNN predictions[15].

For every segmentation mask that cannot be associated with any existing objects, we initialise a new object model whose coordinate frame is centred around the object itself. We back-project all points in the mask into world coordinates and then find the centre and size of these point clouds. To account for possible occlusions, we initialise the TSDF volume size to be 3 times the point cloud size to avoid additional padding. We choose the volume resolution such that each voxel size is slightly bigger than 1mm in order to support detailed object reconstruction. With the octree-based

structure, the unused voxels will not be initialised and the whole system remains memory-efficient. The initial object translation in $\boldsymbol{T}_{WO_n}$ is chosen as the left side corner of the object volume and the orientation is aligned with world coordinates.

F. Raycasting

For raycasting, we use a similar method as proposed in [15]. However, as shown in the pipeline Fig. 2, our system involves at least four raycasting operations: depth rendering in tracking, finding visible objects, IoU calculation, and visualisation, which will be computationally expensive if we continuously raycast all objects each time. To speed up, we raycast all objects only once to find visible objects, and avoid raycasting to invisible objects in the remaining steps on this frame. As in [15], we only raycast voxels whose foreground probability is higher than 0.5. Objects whose voxels are not raycasted at all are considered to be invisible from this view.

V. Experiments

We evaluate our system on a Linux system with an Intel Core i7-7700 CPU at 3.50GHz with 32GB memory. Mask R-CNN segmentation is pre-computed on the GPU using the publicly available weights and implementation [22] without fine-tuning. Each object is stored in a separate octree-based volumetric model, modified based on source code of Supereight [21].

A. Robust camera pose estimation

We first evaluate the camera tracking accuracy in dynamic environments using the widely used TUM RGB-D dataset [23]. The dataset provides RGB-D sequences with ground truth camera trajectory, recorded by a motion capture system. We report the commonly used Root-Mean-Square-Error (RMSE) of the Absolute Trajectory Error (ATE). To evaluate the effect of different cameras motion and environment change conditions, 6 different sequences are investigated. We compare our method with five state-of-the-art dynamic SLAM approaches: joint visual odometry and scene flow (VO-SF) [8], StaticFusion (SF) [9], DynaSLAM (DS) [11], Co-Fusion (CF) [12], and MaskFusion (MF) [14]. VO-SF [8], SF [9], and DS [11] were designed for reconstructing the static background with dynamic parts ignored (or even inpainted as in DS [11]). CF [12] and MF [14] were designed for multi-object reconstruction. In all these methods, DS [11] is the only method using feature-based sparse tracking (not reconstructing moving objects at all), while the remaining ones use dense tracking methods as ours. For fair comparison, we compare first with dense tracking methods and take DS [11] as an additional reference. Table I reports our experimental results.

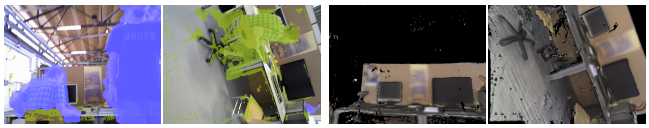
From the table I, we can see that our system achieves best results in almost all sequences among dense tracking method. Our method even outperforms VO-SF and SF, which were designed especially for robust camera tracking in a dynamic environment. Fig. 3 shows two inputs and the reconstruction results in the challenging "f3w halfsphere" sequence. We highlight rejected segmentation masks in the input images, with the geometrically refined mask labelled as human in blue and the high residual regions during motion refinement in green. It can be noted in Fig. 3 that even when Mask R-CNN fails to recognise a person, our combined segmentation using motion refinement can still reject it. DynaSLAM achieves best tracking accuracy in almost all sequences while it is the only sparse feature-based SLAM system among the tested approaches. It shows the potential advantage of feature trackers in dynamic environments. As part of future work, it would be very interesting to combine a feature-based approach with direct dense tracking/mapping methods to further improve camera tracking accuracy and robustness. This could also help to overcome current failure-cases in challenging conditions such as very reflective scenes or fast motions.

TABLE I

Quantitative comparison of camera tracking

Sequence	ATE RMSE (cm)					
	VO-SF	SF	CF	MF	**Ours**	DS*
f3s static	2.9	1.3	1.1	2.1	**1.0**	**-**
f3s xyz	11.1	4.0	**2.7**	3.1	6.2	**1.5**
f3s halfsphere	18.0	4.0	3.6	5.2	**3.1**	**1.7**
f3w static	32.7	**1.4**	55.1	3.5	2.3	**0.6**
f3w xyz	87.4	12.7	69.6	10.4	**6.8**	**1.5**
f3w halfsphere	73.9	39.1	80.3	10.6	**3.8**	**2.5**

*: feature-based sparse approach. The others are dense-tracking approaches.



Input with estimated masks Reconstructed results

Fig. 3. Robust camera tracking and background reconstruction in a dynamic environment (in "f3w halfsphere" sequence). Moving persons are rejected due to the semantic labelling of Mask R-CNN (in blue) or during motion refinement (in green).

B. Object reconstruction evaluation for other components

We also tested our method within a fully controlled synthetic environment using photo-realistic rendering and trajectory simulation [24]. We selected a typical indoor scene with a sofa and a chair being translated and rotated in front of the camera. We implicitly evaluate object pose estimation accuracy via object reconstruction error.

To evaluate the effect of segmentation, we replaced the segmentation pipeline with ground truth masks (G.T. Seg.). We also compared our object-oriented tracker with virtual camera (V.C.) tracking to see if our parametrisation improves tracking accuracy. We further compare with Co-Fusion(CF) [12] using their public code. Table II reports the mean and standard deviation of reconstruction error in these experiments. The results show that our system can achieve more accurate object reconstructions. The difference between using ground truth masks and our own segmentation component is negligible in the specific example. The higher error obtained using virtual camera tracking demonstrates the reliability of our object-centric tracking, especially for large object rotations. Fig. 4 shows the visualisation comparison

TABLE II

OBJECT RECONSTRUCTION ERROR (AVG./STD., IN CM)

Method	CF	Ours with v.c. tracking	**Ours**	Ours with GT-Seg
Sofa	1.72/1.62	1.68/1.90	**0.74/0.79**	0.46/0.59
Chair	1.19/**1.33**	1.13/1.58	**1.00**/1.66	0.92/1.74

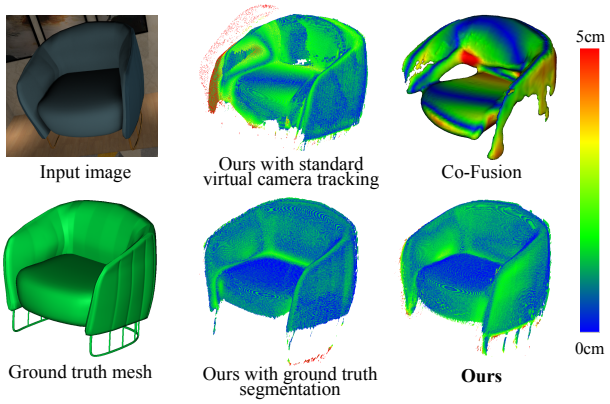


Fig. 4. Comparison of reconstruction error for a moving sofa.

results on the sofa reconstruction.

C. Real-world applications

We demonstrated our proposed method in various scenarios to show its capabilities. Fig. 5 shows the results in two scenes "rotated book" and "cup and bottle". For each input image, we provide label image and reconstruction to show the detailed reconstruction, reliable tracking, and segmentation. With separate volumetric maps for each object, our object models do not collide with each other, which is more suitable for multiple instance SLAM than a surfel-based system. Fig. 6 also shows a scene where our system can simultaneously support the robust tracking of more than 6 moving objects while maintaining a highly detailed reconstruction. As a qualitative comparison, we also show the reconstruction from Co-Fusion, which did not segment and reconstruct these moving objects successfully because the motion was not of sufficient magnitude. In addition, surfel-based systems, such as Co-Fusion and MaskFusion, do not provide the same level of details per object. On the contrary, our system can maintain highly detailed reconstructions and nevertheless keep efficient memory usage thanks to the octree data structure. More results can be seen in the video attachment.

D. Runtime analysis

We evaluated the average computational time for the components of our dynamic SLAM system in different sequences with approximately 3 to 6 objects being moved. Processing time (all on CPU) for each frame averages 400 ms with more than 25 objects being generated in the scene. When a new object is detected, the initialisation takes around 10 ms per object. Tracking time scales mainly with moving objects (MO) while segmentation, integration and raycasting scales with visible objects (VO). A more-detailed breakdown of computation time for each component is shown in Table III.

TABLE III

RUN-TIME ANALYSIS OF SYSTEM COMPONENTS (MS)

Components	Tracking	Segmentation	Integration	Raycasting
Time (ms)	43/MO	10/VO	12/VO.	8/VO

Fig. 5. Qualitative demonstration: input RGB (top row), semantic class prediction (middle row) and geometry reconstruction result (bottom row).

Fig. 6. Qualitative comparison with Co-Fusion: input RGB (left column), our reconstruction results (middle) and Co-Fusion results (right column).

We would like to highlight that our current system only runs on CPU without being highly optimised for performance yet. We believe a high frame-rate version of our system is achievable by exploiting GPU parallelisation.

VI. CONCLUSIONS

We present a novel approach for multi-instance dynamic SLAM using an octree-based volumetric representation. It robustly tracks camera pose in dynamic environment and continuously estimates dense geometry, semantics, and object foreground probabilities. Experimental results in various scenarios demonstrate the effectiveness of our method in indoor environments. We hope our method paves the way for new applications in indoor robotic applications, where an awareness of environment change, free space, and object-level information will empower the next generation of mobile robots.

ACKNOWLEDGMENTS

We wish to thank John McCormac and Emanuele Vespa for fruitful discussions. This research is supported by Imperial College London and the EPSRC grant Aerial ABM EP/N018494/1. Binbin Xu holds a China Scholarship Council-Imperial Scholarship.

REFERENCES

[1] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 29, no. 6, pp. 1052–1067, 2007.

[2] G. Klein and D. W. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[3] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-Time Dense Surface Mapping and Tracking," in *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[4] H. Durrant-Whyte and T. Bailey, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[5] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[6] C.-C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[7] R. A. Newcombe, D. Fox, and S. M. Seitz, "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[8] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, "Fast odometry and scene flow from rgb-d cameras based on geometric clustering," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[9] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "StaticFusion: Background reconstruction for dense rgb-d slam in dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[10] D. Barnes, W. Maddern, G. Pascoe, and I. Posner, "Driven to distraction: Self-supervised distractor learning for robust monocular visual odometry in urban environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[11] B. Bescós, J. M. Fácil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, 2018.

[12] M. Rünz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[13] I. A. Bârsan, P. Liu, M. Pollefeys, and A. Geiger, "Robust dense mapping for large-scale dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[14] M. Rünz and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," *arXiv preprint arXiv:1804.09194*, 2018.

[15] J. McCormac, R. Clark, M. Bloesch, A. J. Davison, and S. Leutenegger, "Fusion++:volumetric object-level slam," in *Proceedings of the International Conference on 3D Vision (3DV)*, 2018.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[17] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," *International Journal of Robotics Research (IJRR)*, vol. 35, no. 14, pp. 1697–1716, 2016.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014, pp. 740–755.

[19] T. Laidlow, M. Bloesch, W. Li, and S. Leutenegger, "Dense RGB-D-Inertial SLAM with map deformations," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2017.

[20] M. Bloesch, H. Sommer, T. Laidlow, M. Burri, G. Nützi, P. Fankhauser, D. Bellicoso, C. Gehring, S. Leutenegger, M. Hutter, and R. Siegwart, "A Primer on the Differential Calculus of 3D Orientations," *CoRR*, vol. abs/1606.0, 2016. [Online]. Available: http://arxiv.org/abs/1606.05285

[21] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. Kelly, and S. Leutenegger, "Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping," *IEEE Robotics and Automation Letters*, 2018.

[22] Y. Wu *et al.*, "Tensorpack," https://github.com/tensorpack/, 2016.

[23] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012.

[24] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger, "Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.